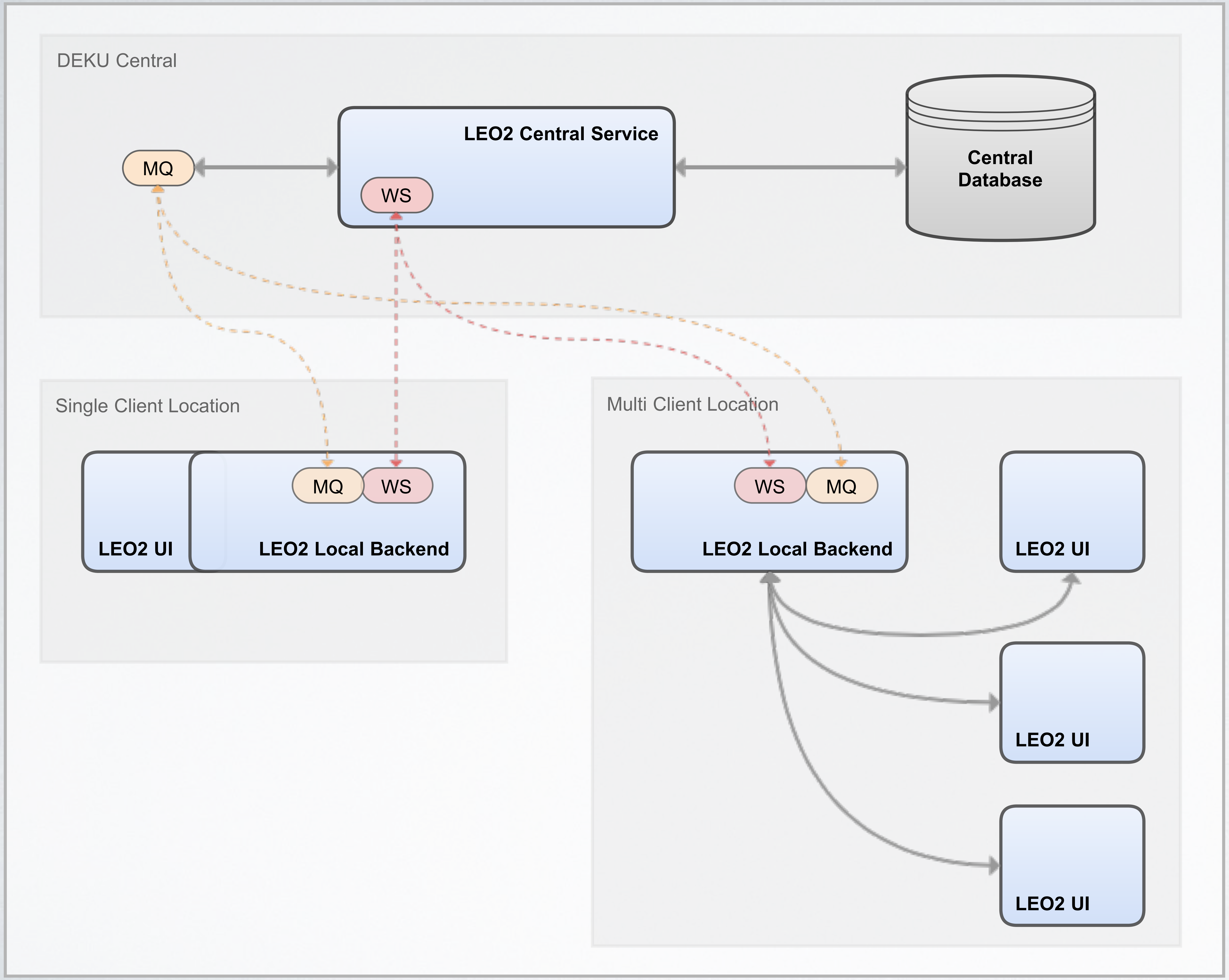
LEOZ

Technicals

Design Considerations

- Decentralised
- Zero-dependency deployment
- Cross-platform / portable
- Non-proprietary
- Small development team

Backend & User Interface



Component Requirements

- Embeddable
- Portable
- Automatic Updates
- Database
- Caching
- Webserver
- Message Bus
- File Synchronisation
- User Interface
- Native Interoperability
- Discovery

Platform Selection

- **.NET/CLR**
  - + Great language for all use cases (C#)
  - − Limited selection of open frameworks & standards
  - − Lack of good cross platform IDE
  - − Lack of suitable cross platform UI frameworks

Platform Selection

- **Virtualization**

+ No constraints

– Complicated updates

– Heavy on resources. Slow on older CPUs

– No user interface (except web)

What about Web

- + Great frameworks for cross platform desktop UI
- + Supports browser based and stand-alone desktop applications
  - Having both does not come for free (yet)
- − Adds another full stack on top of backend
- − Short framework lifecycles
- − Native interoperability is difficult (and ugly)
- − Cumbersome cross browser tests
- − Paradigm change ahead (WebAssembly)

Platform Selection

- **Java/JVM**
  - + Rich ecosystem of open frameworks & standards
  - + Comprehensive cross platform IDEs
  - + Integrated cross platform desktop UI (JavaFX)
  - + Covers Android
  - − <u>Bad language (Java)</u>

Oh no. It’s Java

– Antiquated. Moves (very) slowly

– Incredible amount of boilerplate. Hard to read

– Lack of language features enforces bad design

– No fun. It’s all the little things

– Restrains productivity

+ Fast compiler

JVM Languages

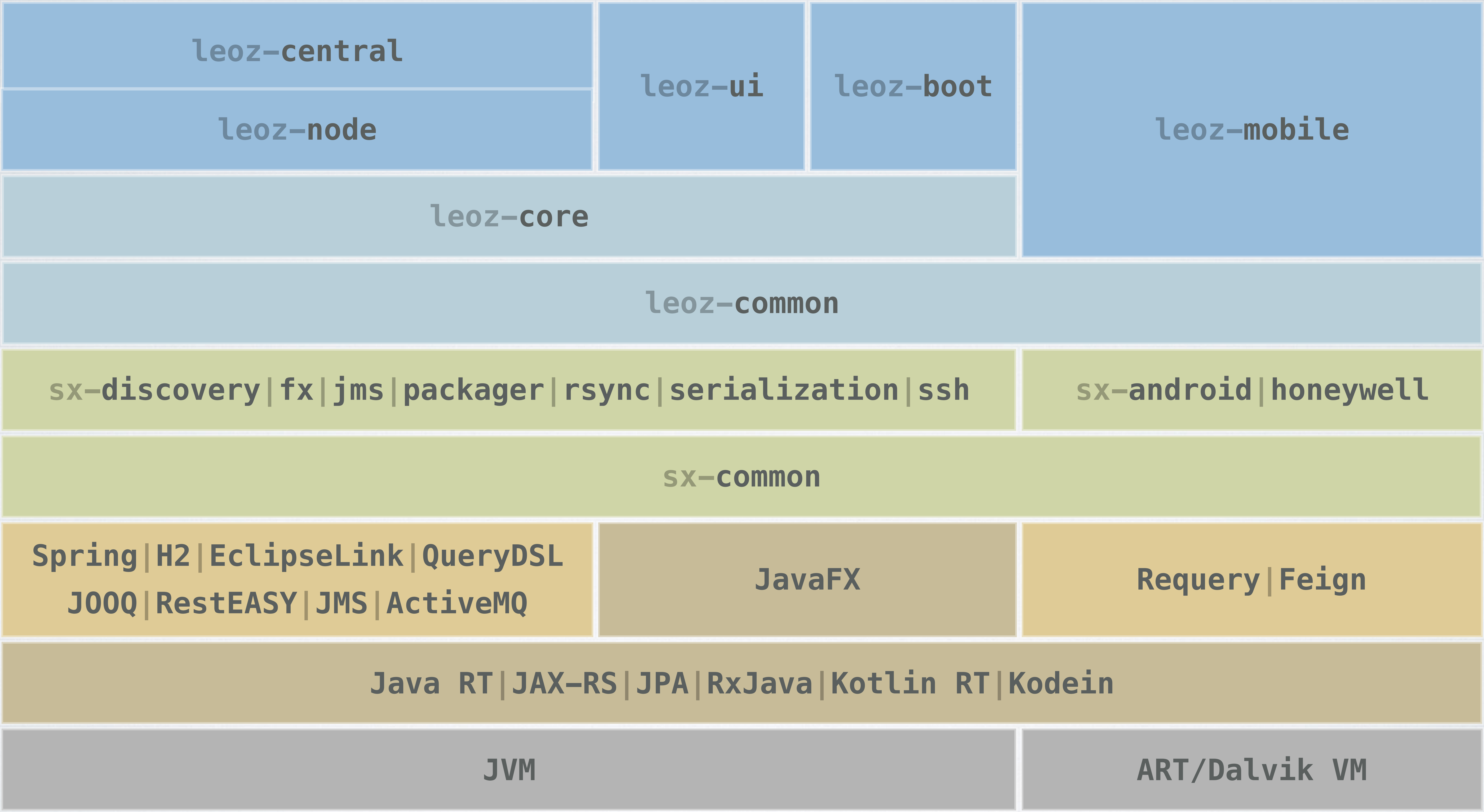
	Scala 2.x	Groovy 2.x	Kotlin 0.x
Modern	✓	✓	✓
Balanced	○	✓	✓
Compilation Speed	○	○	✓
Good Java Interop	○	✓	✓
Android Support	✓	○	✓
Small Runtime Footprint	○	○	✓
IDE Support	✓	✓	✓
Mature	✓	✓	○

Platform Selection

- **Java/JVM**
  - + Rich ecosystem of open frameworks & standards
  - + Comprehensive cross platform IDEs
  - + Integrated cross platform desktop UI (JavaFX)
  - + Covers Android
  - + Great language for all use cases (Kotlin)

LEOZ

Project Structure



leoz-central

leoz-node

leoz-ui

leoz-boot

leoz-mobile

leoz-core

leoz-common

sx-discovery|fx|jms|packager|rsync|serialization|ssh

sx-android|honeywell

sx-common

Spring|H2|EclipseLink|QueryDSL
JOOQ|RestEASY|JMS|ActiveMQ

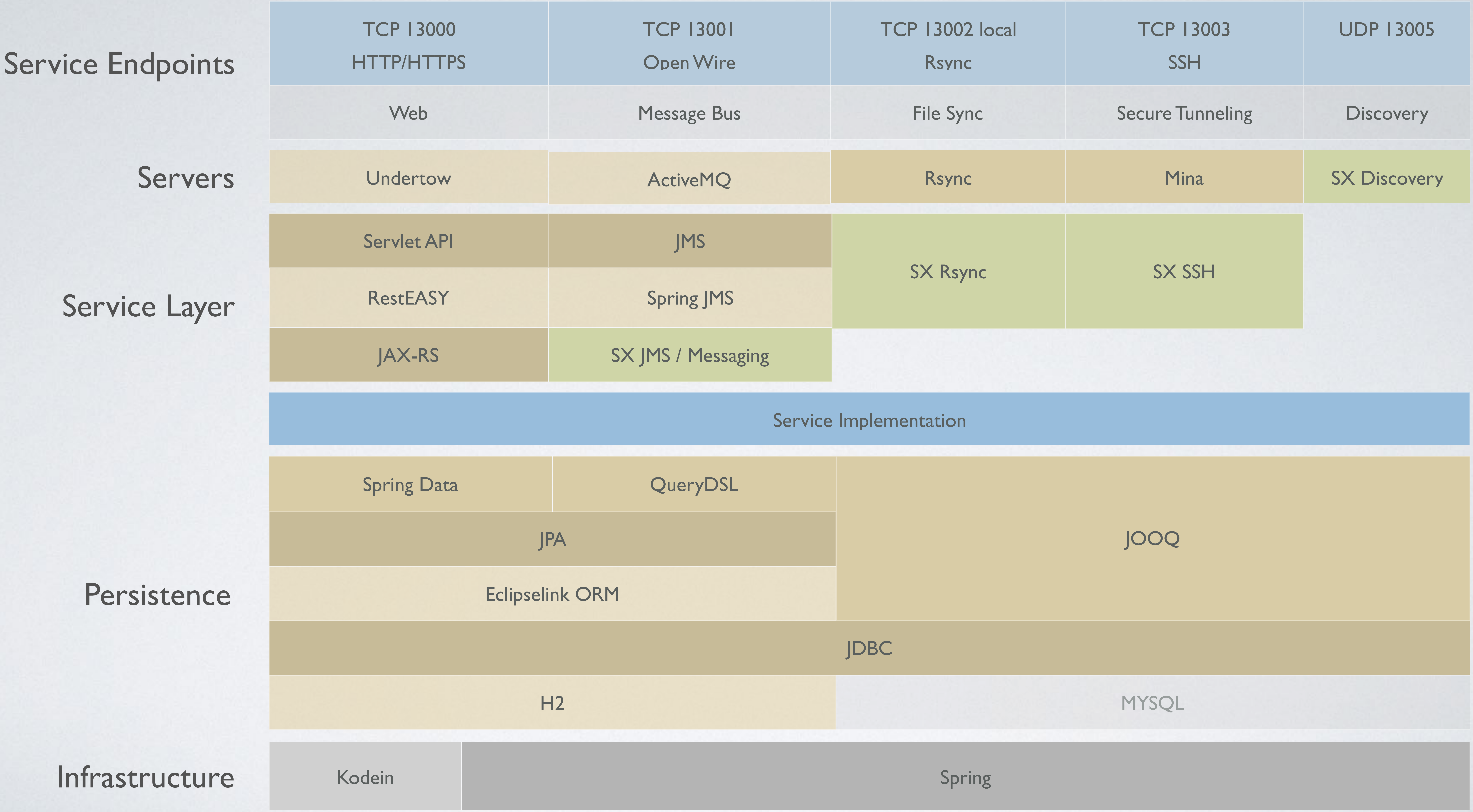
JavaFX

Requery|Feign

Java RT|JAX-RS|JPA|RxJava|Kotlin RT|Kodein

JVM

ART/Dalvik VM



Service Endpoints

TCP 13000 HTTP/HTTPS
TCP 13001 Open Wire
TCP 13002 local Rsync
TCP 13003 SSH
UDP 13005

Web
Message Bus
File Sync
Secure Tunneling
Discovery

Servers

Undertow
ActiveMQ
Rsync
Mina
SX Discovery

Service Layer

Servlet API
JMS
SX Rsync
SX SSH
RestEASY
Spring JMS
JAX-RS
SX JMS / Messaging

Service Implementation

Persistence

Spring Data
QueryDSL
JOOQ
JPA
Eclipselink ORM
JDBC
H2
MYSQL

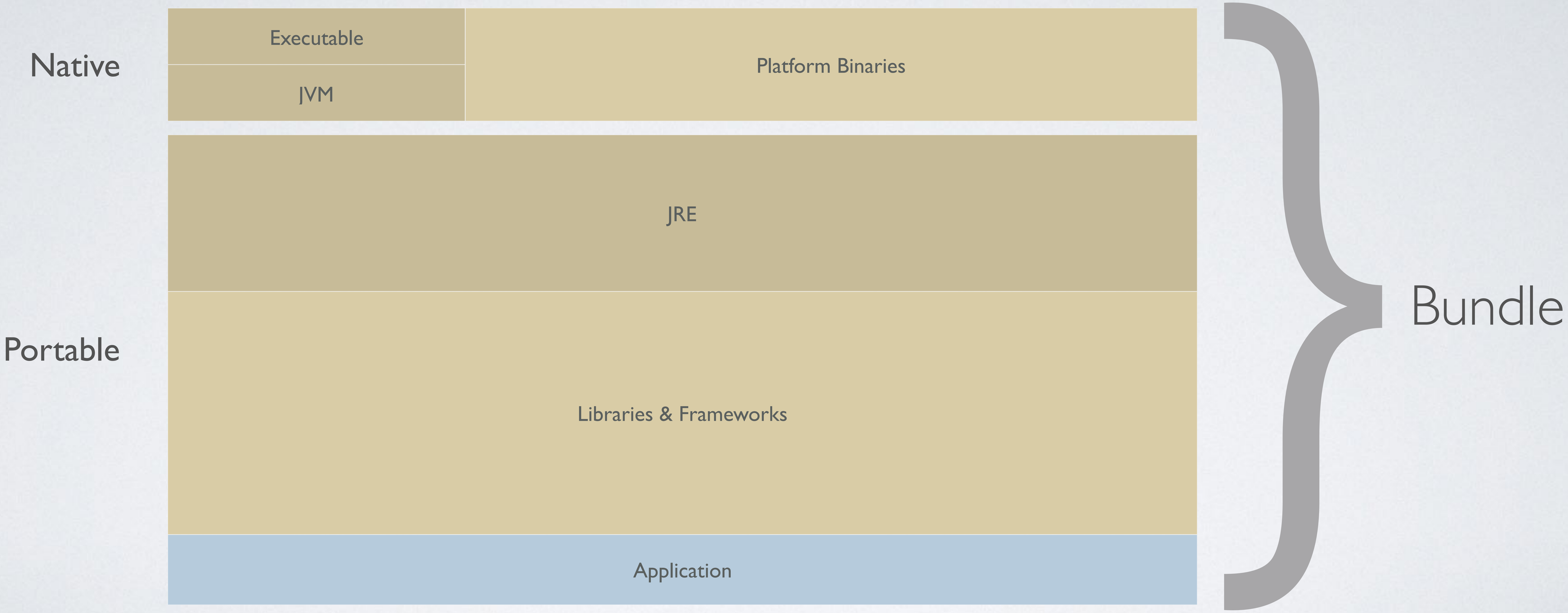
Infrastructure

Kodein
Spring

BUILD TOOLCHAIN

- Gradle
  - Build master. IDEs synchronise.
  - Integrates and automates build related tasks and tools
    - Code generation
    - Compilation
    - Testing
    - Packaging
    - Publishing

Packaging



Native

Executable

JVM

Platform Binaries

Portable

JRE

Libraries & Frameworks

Application

Bundle

Packaging

- Packager Gradle Plugin
  - Leverages Java(FX) Native Packager
  - Leverages Rsync for efficient publishing
  - Cross platform builds (except JDK/JRE is updated)
  - Self-extracting packages (for Windows)

Protocols & Serialisation

- Everything is strong typed
- Versioning & Migrations
- REST
  - Deterministic serialisation
  - Doesn’t require embedded type information
- Message Bus
  - Non-deterministic serialisation
  - Type information must be included
  - Must not break when refactored (can’t use class/package names)

Todo

- A Lot